UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P O Box 1450
Alexandria, Virginia 22313-1450
www uspto gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/734,959	12/12/2003	Tin-Fook Ngai	20002/17846	4798

34431 7590 11/24/2008
HANLEY, FLIGHT & ZIMMERMAN, LLC
150 S. WACKER DRIVE
SUITE 2100
CHICAGO, IL 60606

EXAMINER	
DAO, THUY CHAN	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
11/24/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/734,959	NGAI ET AL.	
	Examiner	Art Unit	
	Thuy Dao	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 August 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,12,14,19,21,23,24,29,31 and 32* is/are pending in the application.

4a) Of the above claim(s) *11,20 and 28* is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,12,14,19,21,23,24,29,31 and 32* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *12 December 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the amendment filed on August 29, 2008.

2. Claims 1-10, 12-19, 21-27, and 29-33 have been examined.

Response to Amendments

3. In the instant amendment, claims 1, 12, 14, 19, 21, 23, 24, 29, 31, and 32 have been amended; claims 11, 20, and 28 have been canceled.

Response to Arguments

4. Applicants' arguments have been considered.

a) The limitations at issue "*determining a misspeculation cost value for at least one of the speculative parallel thread candidates; selecting a set of speculative parallel threads from the set of speculative parallel thread candidates based on the misspeculation cost value*" (e.g., claim 1, lines 3-6; Remarks, pp. 10-11).

The examiner respectfully disagrees with Applicants' assertions. Kim explicitly discloses:

determining a misspeculation cost value for at least one of the speculative parallel thread candidates (e.g., page 4, selecting implicit/speculative threads; page 5, Section 4, speculative storage overflow as a misspeculation cost; determining a speculative storage built up value against the available 4K direct-mapped cache as determining a misspeculation cost value; pp. 9-10, implicit/speculative thread selection based on speculative storage overflow value, i.e., determining a built up value either below 4K direct-mapped cache (not overflow) or more than (overflow) 4K direct-mapped cache)

selecting a set of speculative parallel threads from the set of speculative parallel thread candidates based on the misspeculation cost value (e.g., page 10, first paragraph, the misspeculation cost value/speculative storage overflow happens when all speculatively written values built up in the course of the thread execution exceeds the capacity of the speculative storage such as 4K direct-mapped cache (page 11) or

16K direct-mapped cache (page 12); page 10, fourth paragraph, implicit/speculative thread selection based on speculative storage overflow).

b) The examiner notes that the Applicants acknowledged that Kim indeed teaches implicit/speculative thread and implicit/speculative thread is distinct with explicit thread "...Also, because Kim et al. only consider fully parallel loops for explicit threading (see Section 2.2, page 4, last paragraph), the explicit threading described by Kim et al. is not speculative and only the implicit loops are considered by Kim et al. to be speculative. This distinction between explicit and implicit (speculative) threading is also generally known in the art" (Remarks, page 11, first paragraph, emphasis added).

Accordingly, Applicants' arguments regarding explicit threads and their selection based on the "outermost parallel loop" (page 11, beginning of first paragraph), "doubly nested", "workload" (page 11, beginning of second paragraph), "inner-loop serial loops", "function calls" (beginning of page 12) have not provided any proof/evidence to show that Kim does not teach "*determining a misspeculation cost value for at least one of the speculative parallel thread candidates; selecting a set of speculative parallel threads from the set of speculative parallel thread candidates based on the misspeculation cost value*" as claimed (emphasis added).

c) The examiner further notes that the Applicants acknowledged that Kim describes "explicitly multithread the code or [multithread the code] implicitly (speculatively)" (Remarks, page 11, emphasis added). Applicants further asserted,

> "...Determining the presence of inner-loop serial loops ensures a processor will not stall or hang while executing the program due to dependencies on variables that span multiple threads, and cannot be fairly construed as determining a misspeculation cost. Third, if the loop L contains any function calls, the loop L is executed as an explicit thread because implicit execution of the thread by a processor would incur stalls and hangs due to artificial dependencies of the speculative thread. Determining function calls also does not constitute

> determining a misspeculation cost. Finally, if a thread would incur speculative storage overflow (i.e., if the number of speculatively written values exceed the capacity of the speculative storage in the L1 cache) by executing an implicit thread, the loop L is executed explicitly. (See Section 4, page 10, paragraphs 2 and 3). Thus, none of the foregoing criteria described by Kim et al. could be interpreted to be a misspeculation cost as recited in claim 1" (emphasis added).

As acknowledged above by the Applicant, "speculative storage overflow" is a misspeculation cost, so that when the "speculative storage overflow"/misspeculation cost happens, the selection between explicit thread and implicit/speculative thread is decided (in this case, no implicit/speculative thread is selected based on the happened "speculative storage overflow"/misspeculation cost).

Furthermore, as set forth in paragraph (a) above, Kim explicitly at least teaches determining "speculative storage overflow" is determining a misspeculation cost value.

d) The limitations at issue "*identifying a data dependency violation in a set of speculative parallel thread candidates; determining a likelihood that the data dependency violation will occur; determining an amount of computation required to recover from the data dependency*" (claim 33, lines 2-6; Remarks, pp. 13-14).

The examiner respectfully disagrees with Applicants' assertions. Kim explicitly teaches:

identifying a data dependency violation in a set of speculative parallel thread candidates (see for example pages 6-7, "The second type of dependence .... "; page 10, Loop attribute);

determining a likelihood that the data dependency violation will occur (see for example pages 9-10, "Implicit/Explicit Thread Selection"; Loop attribute, when there is an inner, serial loop with cross-iteration dependences, then there is a likehood that

the data dependency violation will occur; when the candidate loop contains function calls, then there is a likehood that the data dependency violation will occur);

determining an amount of computation required to recover from the data dependency (see for example page 5, "Two-version Code for Explicit/Implicit Threading"; page 10, the number of instruction cycles to recover from the data dependency, in which the amount can reach several hundred instruction cycles)

selecting at least one of the set of speculative parallel thread candidates based on a lowest likelihood of misspeculation (see for example pages 9-10, "Implicit/Explicit Thread Selection"; if the candidate loop does not have any inner, serial loop (a lowest likehood), then it can be selected as a speculative thread; if the candidate loop does not have function calls (a lowest likehood), then it can be selected as a speculative thread).

Accordingly, Applicants' arguments are not persuasive. The examiner respectfully maintains ground of the 35 USC §102(b) rejection over claims 1-10, 12-19, 21-27, and 29-33 as being anticipated by Kim.

Claim Rejections – 35 USC §102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1-10, 12-19, 21-27, and 29-33 are rejected under 35 U.S.C. 102(b) as being anticipated by Kim (art of record, "The Structure of a Compiler for Explicit and Implicit Parallelism").

Claim 1:

Kim discloses a method of compiling a program comprising:

identifying a set of speculative parallel thread candidates (see for example page 2, "Recognition of parallelism");

determining a misspeculation cost value for at least one of the speculative parallel thread candidates (e.g., page 4, selecting implicit/speculative threads; page 5, pointing to Section 4, speculative storage overflow as a misspeculation cost; checking speculative storage built up value against available 4K direct-mapped cache as checking a misspeculation cost value; pp. 9-10, Section 4 implicit/speculative thread selection based on speculative storage overflow value, i.e., either below 4K direct-mapped cache (not overflow) or more than (overflow) 4K direct-mapped cache)

selecting a set of speculative parallel threads from the set of speculative parallel thread candidates based on the misspeculation cost value (e.g., page 10, first paragraph, the misspeculation cost value/speculative storage overflow happens when all speculatively written values built up in the course of the thread execution exceeds the capacity of the speculative storage such as 4K direct-mapped cache (page 11) or 16K direct-mapped cache (page 12); page 10, fourth paragraph, implicit/speculative thread selection based on speculative storage overflow); and

generating program code based on the set of speculative parallel threads (see for example page 2, "Thread code generation").

Claim 2:

Kim further discloses a method as defined in claim 1 wherein identifying the set of speculative parallel thread candidates comprises identifying program regions (see for example page 2, "Selecting explicit and implicit threads").

Claim 3:

Kim further discloses a method as defined in claim 1 wherein at least one of the speculative parallel thread candidates comprises at least one program region (see for example page 2, "Selecting explicit and implicit threads").

Claim 4:

Kim further discloses a method as defined in claim 1 wherein at least one of the speculative parallel threads comprises at least one program region (see for example page 2, "Selecting explicit and implicit threads").

Claim 5:

Kim further discloses a method as defined in claim 1 wherein identifying the set of speculative parallel thread candidates comprises identifying program loops (see for example pages 4-5, "2.2 Thread Selection").

Claim 6:

Kim further discloses a method as defined in claim 1 wherein at least one of the speculative parallel thread candidates comprises a program loop (see for example pages 4-5, "2.2 Thread Selection").

Claim 7:

Kim further discloses a method as defined in claim 1 wherein at least one of the speculative parallel threads comprises a program loop (see for example pages 5-6, "2.3 Thread Preprocessing").

Claim 8:

Kim further discloses a method as defined in claim 1 wherein identifying the set of speculative parallel thread candidates comprises identifying a main thread (see for example pages 5-6, "2.3 Thread Preprocessing").

Claim 9:

Kim further discloses a method as defined in claim 8 wherein the main thread comprises a current iteration of a program loop, and the speculative parallel thread candidate comprises a next iteration of the same program loop (see for example pages 5-6, "2.3 Thread Preprocessing").

Claim 10:

Kim further discloses a method as defined in claim 8 wherein the main thread comprises a current iteration of a program loop, and the speculative parallel thread comprises a next iteration of the same program (see for example pages 5-6, "2.3 Thread Preprocessing").

Claim 12:

Kim further discloses a method as defined in claim 1 wherein determining the misspeculation cost value comprises:

identifying a data dependency in at least one of the speculative parallel thread candidates (see for example page 6, paragraph 2, "The first type of P dependence .... ");

determining, for the data dependency, a likelihood that a dependency violation will occur; and determining an amount of computation required to recover from the data dependency violation (see for example page 6, paragraph 2).

Claim 13:

Kim further discloses a method as defined in claim 1 further comprising determining at least one of the following for at least one of the speculative parallel thread candidates: a Size of the speculative parallel thread candidate; and a likelihood representative of the speculative parallel thread candidate (see for example page 6, Fig. 2, and related text).

Claim 14:

Kim further discloses a method as defined in claim 1 wherein at least one of the speculative parallel thread candidates is transformed prior to determining the misspeculation cost value for the at least one of the speculative parallel thread candidates (see for example page 6, Fig. 2, "Code transformation...", and related text).

Claim 15:

Kim further discloses a method as defined in claim 14 wherein the at least one of the speculative parallel thread candidates is transformed by a code reordering (see for example page 6, Fig. 2, "Code transformation...", and related text).

Claim 16:

Kim further discloses a method as defined in claim 14 further comprising determining at least one of the following for at least one of the speculative parallel thread candidates:

a size of the speculative parallel thread candidate; a likelihood representative of the speculative parallel thread candidate (see for example page 6, Fig. 2, and related text); and

a description of the transformation performed on the speculative parallel thread candidate (see for example page 6, Fig. 2, and related text).

Claim 17:

Kim further discloses a method as defined in claim 1 wherein at least one of the speculative parallel threads is transformed prior to code generation (see for example page 6, Fig. 2, "Code transformation...", and related text).

Claim 18:

Kim further discloses a method as described in claim 17 wherein the at least one of the speculative parallel threads is transformed by code reordering (see for example page 6, Fig. 2, "Code transformation...", and related text).

Claim 19:

This is the article version of the claimed method discussed above (Claim 1), wherein all claim limitations have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also anticipated by Kim.

Claim 21:

This is the article version of the claimed method discussed above (Claim 19), wherein all claim limitations have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also anticipated by Kim.

Claim 22:

This is the article version of the claimed method discussed above (Claim 13), wherein all claim limitations have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also anticipated by Kim.

Claim 23:

This is the article version of the claimed method discussed above (Claim 14), wherein all claim limitations have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also anticipated by Kim

Claim 24:

Kim discloses an apparatus (see for example Fig. 1, and related text) to compile a program comprising:

a candidate identifier to identify a set of speculative parallel thread candidates (see for example page 2, "Recognition of parallelism");

a metric estimator to determine a cost value for at least one of the speculative parallel thread candidates (see for example page 6, Fig. 2, and related text);

a speculative parallel thread selector, to select a set of speculative parallel threads from the set of speculative parallel thread candidates based on the misspeculation cost value (see for example page 2, "Selecting explicit and implicit threads", and related text); and

a code generator to generate program code based on the set of speculative parallel threads (see for example page 4, Fig. 1, "Code generator", and related text).

Claim 25:

Kim further discloses an apparatus as defined in claim 24 wherein the candidate identifier comprises a region identifier to identify program regions (see for example page 2, "Recognition of parallelism").

Claim 26:

Kim further discloses a apparatus as defined in claim 24 wherein the candidate identifier comprises a loop identifier to identify program loops (see for example pages 4-5, "2.2 Thread Selection").

Claim 27:

Kim further discloses a apparatus as defined in claim 24 wherein the candidate identifier comprises a candidate selector to select a first one of a program region and a program loop iteration to execute in a main thread, and to select a second one of a program region and a program loop iteration to execute in a speculative parallel thread (see for example pages 4-5, "2.2 Thread Selection").

Claim 29:

Kim further discloses: an apparatus as defined in claim 24 wherein the metric estimator comprises:

a data dependency identifier to identify a data dependency in the speculative parallel thread candidates (see for example page 6, paragraph 2, "The first type of dependence .... ");

a likelihood evaluator to determine a likelihood that a dependency violation will occur (see for example page 6, paragraph 2, "The first type of dependence .... "); and

a recovery size calculator to determine an amount of computation required to recover from the data dependency violation (see for example page 6, Fig. 2, and related text).

Claim 30:

Kim further discloses an apparatus as defined in claim 24 wherein the candidate identifier determines at least one of the following for at least one of the speculative parallel thread candidates:

a size of the speculative parallel thread candidate violation (see for example page 6, Fig. 2, and related text); and

a likelihood representative of the speculative parallel thread candidate violation (see for example page 6, Fig. 2, and related text).

Claim 31:

Kim discloses a system (see for example page 4, Fig. 1, and related text) to compile a program comprising:

a candidate identifier to identify a set of speculative parallel thread candidates (see for example page 2, "Recognition of parallelism");

a metric estimator to determine a cost value for at least one of the speculative parallel thread candidates (see for example page 6, Fig. 2, and related text);

a speculative parallel thread selector, to " select a set of speculative parallel threads from the set of speculative parallel thread candidates based on the misspeculation cost value (see for example page 2, "Selecting explicit and implicit threads", and related text); and

a code generator to generate program code based on the set of speculative parallel threads (see for example page 4, Fig. 1, "Code generator", and related text).

Claim 32:

Kim further discloses a system as define in claim 31 wherein the metric estimator comprises:

a data dependency identifier to identify a data dependency in at least one of the speculative parallel thread candidates (see for example page 6, paragraph 2, "The first type of dependence .... ");

a likelihood evaluator to determine a likelihood that a dependency violation will occur (see for example page 6, paragraph 2, "The first type of dependence .... "); and

a recovery size calculator to determine an amount of computation required to recover from the data dependency violation (see for example page 6, Fig. 2, and related text).

Claim 33:

Kim discloses a method of compiling a program comprising:

identifying a data dependency violation in a set of speculative parallel thread candidates (see for example pages 6-7, "The second type of dependence .... "; page 10, Loop attribute);

determining a likelihood that the data dependency violation will occur (see for example pages 9-10, "Implicit/Explicit Thread Selection"; Loop attribute, when there is an inner, serial loop with cross-iteration dependences, then there is a likehood that the data dependency violation will occur; when the candidate loop contains function calls, then there is a likehood that the data dependency violation will occur);

determining an amount of computation required to recover from the data dependency (see for example page 5, "Two-version Code for Explicit/Implicit Threading"; page 10, the number of instruction cycles to recover from the data dependency, in which the amount can reach several hundred instruction cycles)

selecting at least one of the set of speculative parallel thread candidates based on a lowest likelihood of misspeculation (see for example pages 9-10, "Implicit/Explicit Thread Selection"; if the candidate loop does not have any inner, serial loop (a lowest likehood), then it can be selected as a speculative thread; if the candidate loop does not have function calls (a lowest likehood), then it can be selected as a speculative thread).

Conclusion

7. **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not

mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

8. Any inquiry concerning this communication should be directed to examiner Thuy Dao (Twee), whose telephone/fax numbers are (571) 272 8570 and (571) 273 8570, respectively. The examiner can normally be reached on every Tuesday, Thursday, and Friday from 6:00AM to 6:00PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam, can be reached at (571) 272 3695.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273 8300.

Any inquiry of a general nature of relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is (571) 272 2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Thuy Dao/
Examiner, Art Unit 2192

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192